IMENTATION PAGE

2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
March 1994	Scientific Paper

4. TITLE AND SUBTITLE
Scene Rendering for the Smart Weapons Operability Enhancement Program

5. FUNDING NUMBERS

6. AUTHOR(S)
Luke A. Catania

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Topographic Engineering Center
ATTN: CETEC-PAO
7701 Telegraph Road
Alexandria, VA 22310-3864

8. PERFORMING ORGANIZATION REPORT NUMBER
R-216

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release;
distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

During FY89, the U.S. Army Topographic Engineering Center (TEC), previously the U.S. Army Engineer Topographic Laboratories (USAETL), became a key participant in the Smart Weapons Operability Enhancement (SWOE) program. This program is a tri-service partnership, organized for and working to provide the smart weapons/automatic target recognition (ATR) designers, developers, testers, and evaluators with the integrated information, measurements, modeling and simulation tools necessary to consider and exploit the operational battlefield environment. As part of the SWOE team, TEC is responsible for the rendering module of the "SWOE PROCESS".

The preliminary SWOE PROCESS consisted of receiving output from the SWOE thermal and radiance models, reformatting the output so that the TEC Computer Image Generation (CIG) system could render a simulated infrared (IR) scene. This CIG system was originally developed by Boeing Aerospace, with the software residing on a Gould 32/67 computer under the Mapped Programming Executive (MPX) operating system. During FY91, TEC was responsible for converting selected modules of the MPX-CIG software to Unix, as well as incorporating/integrating the SWOE thermal, radiance, and rendering modules onto a common Unix platform, a Stardent Titan 3040 mini-supercomputer.

14. SUBJECT TERMS
Smart Weapons, Computer Image Generation, infared rendering

15. NUMBER OF PAGES
10

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	

Scene rendering for the
Smart Weapons Operability Enhancement Program

Luke A. Catania

U.S. Army Topographic Engineering Center
Fort Belvoir, Virginia 22060-5546

ABSTRACT

During FY89, the U.S. Army Topographic Engineering Center (TEC), previously the U.S. Army Engineer Topographic Laboratories (USAETL), became a key participant in the [redacted] Operability Enhancement (SWOE) program. This program is a tri-service partnership, organized for and working to provide the smart weapons/automatic target recognition (ATR) designers, developers, testers, and evaluators with the integrated information, measurements, modeling and simulation tools necessary to consider and exploit the operational battlefield environment. As part of the SWOE team, TEC is responsible for the rendering module of the "SWOE PROCESS".

The preliminary SWOE PROCESS consisted of receiving output from the SWOE thermal and radiance models, reformatting the output so that the TEC [redacted] (CIG) system could render a simulated [redacted] (IR) scene. This CIG system was originally developed by Boeing Aerospace, with the software residing on a Gould 32/67 computer under the Mapped Programming Executive (MPX) operating system. During FY91, TEC was responsible for converting selected modules of the MPX-CIG software to Unix, as well as incorporating/integrating the SWOE thermal, radiance, and [redacted] modules onto a common Unix platform, a Stardent Titan 3040 mini-supercomputer.

The process for generating these IR scenes consists of combining 3-D gridded elevation data with a 2-D raster radiance map which is a gray-scale representation of the various surface materials existing for the area. Such surface materials may be medium vegetation and bare ground for example. This "colored" elevation surface is then textured with empirical texture maps representing the various surface materials. Trees and targets are also placed in the scenes. Geographic position information is specified in an image input file along with sensor location, field of view (FOV), sun zenith and azimuth. The image input file is used as input to the image generation portion of the rendering module to create perspective views from the gridded terrain and polygonized models using a Z-buffer algorithm to eliminate hidden surfaces. This paper will detail the process of the rendering module of the SWOE PROCESS.

1. HISTORY

The Computer Image Generation system initially used by TEC for the SWOE Process was developed under a program sponsored by the Defense Advanced Research Projects Agency (DARPA). The system consisted of software to generate static scenes and specialized hardware for real-time simulation. The software was developed on a Gould 32/67 computer running the MPX operating system and was written in FORTRAN with assembly language routines to provide fast input/output (I/O). The hardware system consists of three racks of special purpose hardware which allows real-time fly-thrus of terrain data at the rate of 10000 triangles at 30 frames per second. The process of building a database by combining several forms of input data and generating static scenes using the software system will be discussed in this paper.

2. APPLICATION

The CIG system gives a user the ability to combine real elevation data, color feature maps, models, and texture to create realistic terrain scenes. The application that TEC is currently using this system for is to generate IR images for smart weapons sensor testing. The problem with smart weapons systems is that testing of the weapons cannot

be done for all environmental conditions for all times of the day. The SWOE program leverages technology base studies by Army, Navy, Marine, Air Force and DARPA. The major thrust of the SWOE approach is the integration of measurements, information bases, physics-based models, and state of the art scene rendering techniques to generate synthetic IR scenes. As part of the SWOE program, a primary task was to generate two scenes for a specified site, Fort Hunter-Liggett, California, representing 20 September 1989 at 0600 and 1500.

3. DATA INPUTS

TEC was given all the necessary data for input into the CIG system. This data consisted of elevation data, feature data, radiances for each feature polygon, radiometric texture maps, an M60 tank, an M113 Armored Personnel Carrier (APC), and a tree. TEC was supplied with two sets of data with radiances calculated for 0600 and for 1500.

3.1. Elevation data

The elevation data was supplied in the CIG grid file format which consists of two files, a header file and a data file. The header file specifies the number of elevation grid points in x and y, latitudinal and longitudinal grid spacing specified in arcseconds, the southwest corner of the terrain region, and a brief description. The data file consists of elevation data stored as floating point numbers in column major format from south to north and east to west. A top down shaded relief image of the elevation data used is shown in Figure 1.

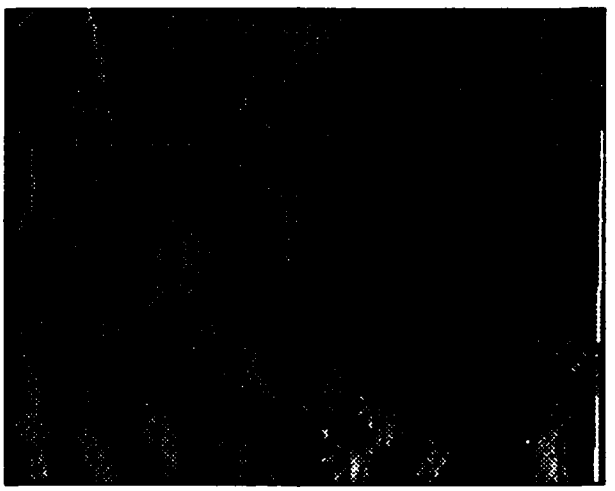

Figure 1. Top down shaded relief image of Fort Hunter-Liggett, California.

3.2. Feature data

The feature data was digitized from existing maps at 2 meter resolution to match the elevation data resolution. This data was supplied in the CIG color file format. This format consists a 512 byte header containing information such as the number of rows and columns, the number of intensities and bits, and northwest and southeast corner. The data contains three planes of two byte data values representing feature identification numbers (FID), surface material categories (SMC), and a filler plane, with the number of values in each plane being the number of rows times the number of columns. An image representing the feature data used for the 0600 and 1500 scene is shown in Figure 2.

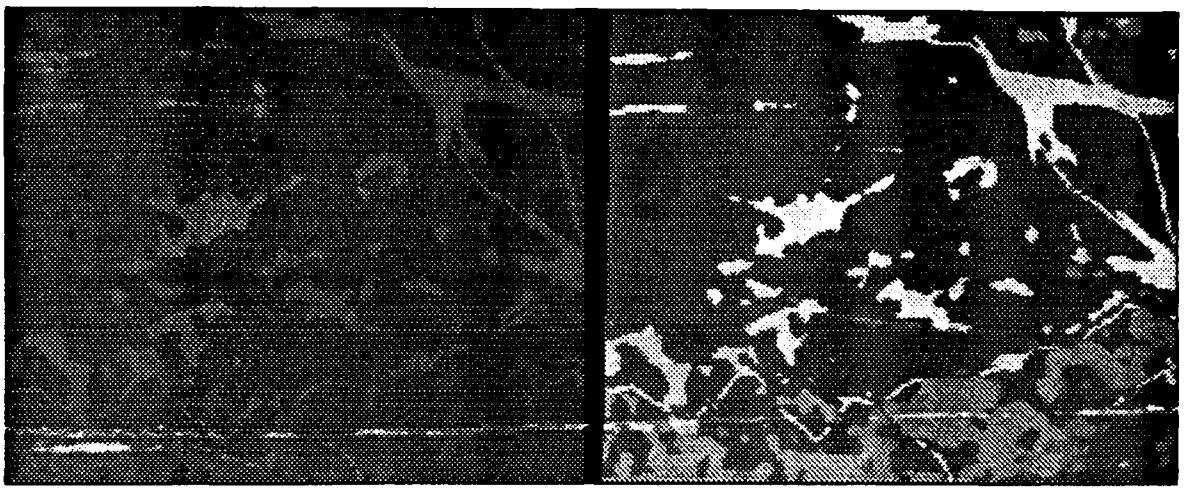

Figure 2. Top down radiance maps representing 1225 feature polygons at 0600 and 1500.

3.3. Feature radiances

The radiance data for each terrain feature polygon was calculated by using data output from the Interim Thermal Model (ITM) that was piped into the Improved Background Radiance Model (IBRM). The output of the IBRM program consisted of one radiance per terrain feature polygon measured in watts/cm^2. Each of these values were associated with an FID in the feature file discussed in the previous section.

3.4. Texture maps

Texture maps are images used to enhance the detail of terrain or models. In other words, rather than having a high resolution database, texture can be used to make the terrain appear high resolution. The texture maps supplied were calculated from empirical data, that is, data used to generate the maps are based on observation and not theory. The texture maps were supplied in the CIG sector image format with a size of 256x256 pixels and represented three feature types: bare soil, medium vegetation, and forest canopy. The sector image format contains the same fields in its header as the color file format discussed in section 3.2. The data following the 512 byte header contains one plane of one byte pixel information since the data are monochrome. The texture maps were produced by a program written in turbo pascal running on an IBM compatible personal computer. The program calculated texture maps for 20 September 1989 at 0600 and 1500. The texture images representing the three surface material categories are shown in Figure 3.

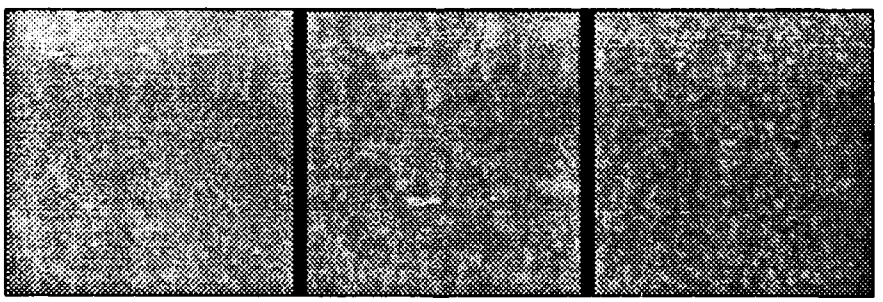

Figure 3. Texture maps representing bare soil, medium vegetation and, forest canopy at 1500 are illustrated from left to right.

3.5. Models

The models that were supplied consisted of an M60 tank, an M113 APC, and a tree. The tree model that was delivered had it's branch structure modeled after an actual tree measured in the field. There was no model for the leaf structure, so a function to randomly place triangulated leaves on the branches was used. The number of leaves place on the trees is based upon the season of the year. That is, winter would have no leaves and summer would be full of leaves and spring and fall would have some variation between. The 1500 hour tree had a shadow pasted to the bottom of it. That is, the shadow is physically part of the geometry data and does not interact with the terrain at all. This noninteraction caused the trees to cut through the terrain or appear above the terrain in some cases. The M60 and M113 geometry's were obtained from the Ballistic Research Laboratory (BRL). The M60 tank at 1500 also came with a shadow as part of it's geometry and had the same problem as the tree model in it's non-interaction with the terrain. All the models were supplied in the CIG Vertex and Polygon (VP) format. The vertex file contained x,y,z model vertices and radiance values for each vertex. The polygon file contained the number of sides in the polygon followed by indices into the vertex file. This data would be repeated in the file for each polygon in the model. The radiances are repeated three times in the file for compatibility with original VP files used in the CIG system. This format is depicted in Table 1.

Vertex File	Polygon File
$X_0\ Y_0\ Z_0\ R_0\ R_0\ R_0$	3 (number of sides)
$X_1\ Y_1\ Z_1\ R_1\ R_1\ R_1$	index to vertex 1
$X_2\ Y_2\ Z_2\ R_2\ R_2\ R_2$	index to vertex 2
$X_n\ Y_n\ Z_n\ R_n\ R_n\ R_n$	index to vertex 3

Table 1. Vertex and polygon model file format.

The indexes for the vertices were defined according to the right-hand rule or in a counter-clockwise order. This is necessary for calculating surface normals correctly for polygon coloring. If the vertices are not defined in this order, the surface normals of the polygons will point into the model instead of out which will result in improper rendering. The models described in this section are depicted in Figure 4.

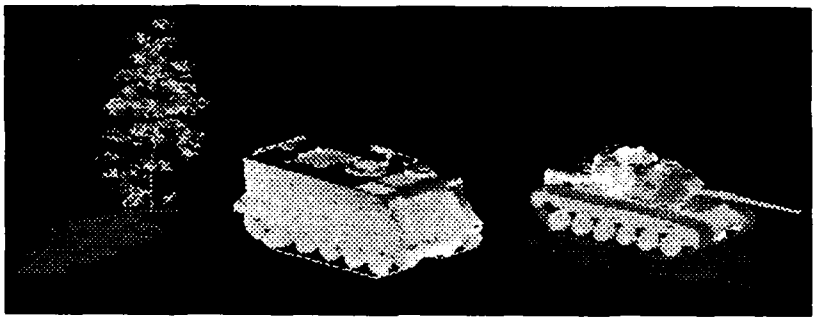

Figure 4. A tree model, M113 APC, and M60 tank with radiances calculated for 1500 hours.

3.6. Radiance to grayscale mapping

Since the CIG could not process floating point radiance values contained in the feature radiance and model files, these radiances had to be converted to gray scale colors in the 0 to 255 range. The conversion of the radiances to gray scale was a simple linear mapping. A routine was written to read in the terrain feature radiances, tree radiances, and target radiances and calculate the absolute minimum and absolute maximum radiances. The minimum radiance was mapped to zero and the maximum was mapped to 255. A binning process was performed in which 256 radiance bins were made to map the rest of the radiances. This bin size was calculated simply by dividing the difference of the maximum and minimum by 256. The model output of this program is the vertex file with the radiances replaced by grayscale values. The feature radiance file output is modified by adding an additional column containing

grayscale values for each feature polygon. The feature radiance file is then reformatted into a CIG color Look-Up Table (LUT) for use in scene rendering.

4. Software integration

The initial Fort Hunter-Liggett IR scenes were generated using a nonintegrated process. Two scenes, representing 20 September 1989 at 0600 and 1500, were generated for the Fort Hunter-Liggett site. For the initial Hunter-Liggett scenes, the thermal calculations were run on a SUN workstation, the radiances were run on an Silicon Graphics workstation, and the images were rendered on a Gould.

Over a year period, the CIG system was ported to a Unix platform, a Stardent Titan 3040 mini-supercomputer. The thermal and radiance models were also ported to the Titan with little difficulty since they already existed on Unix based machines. Only portions of the CIG system used to generate the initial IR scenes were ported to the Unix platform. This task was expected to be somewhat difficult because of major differences in the two machines, the Gould running MPX and the Titan running Unix.

The first step in porting the code was to physically bring over the necessary portions of the code to the Titan from the Gould. This was done through a combination of ethernet connection and a 9-track tape drive. Several problems that were encountered in porting the code were incompatibilities in the FORTRAN programming language due to extensions, inline assembly language and assembly language calls, and systems calls. The code was compiled to determine where all these problems would occur Some of the system calls that the Gould used were to get extended memory, system date and time. There was no need to get extended memory on the Titan due to it's 256 megabytes of base memory compared to the Gould's 16 megabytes. System date and time routines were replaced with Unix system routines. The assembly language calls were mostly to provide fast I/O. These calls were replaced with calls to C language routines. The inline assembly language was also replaced by calls to C routines.

In order to properly test the code, it was ported in the order that it had to be executed for perspective scene generation. That is, common routines that were used by all programs were ported first, followed by the programs: shaded relief, terrain database creation, terrain coloring, terrain texturing, model creation, image input file creation, and image generation. All these programs and their inputs are described in the next section and will step through the process of creating, coloring, and texturing a database to the final process of how the image generator uses world to screen transformations, polygonal tiling, and z-depth buffer calculations to create a final perspective image. The perspective scenes that were generated for 20 September 1989 at 0600 and 1500 are depicted in Figure 5.

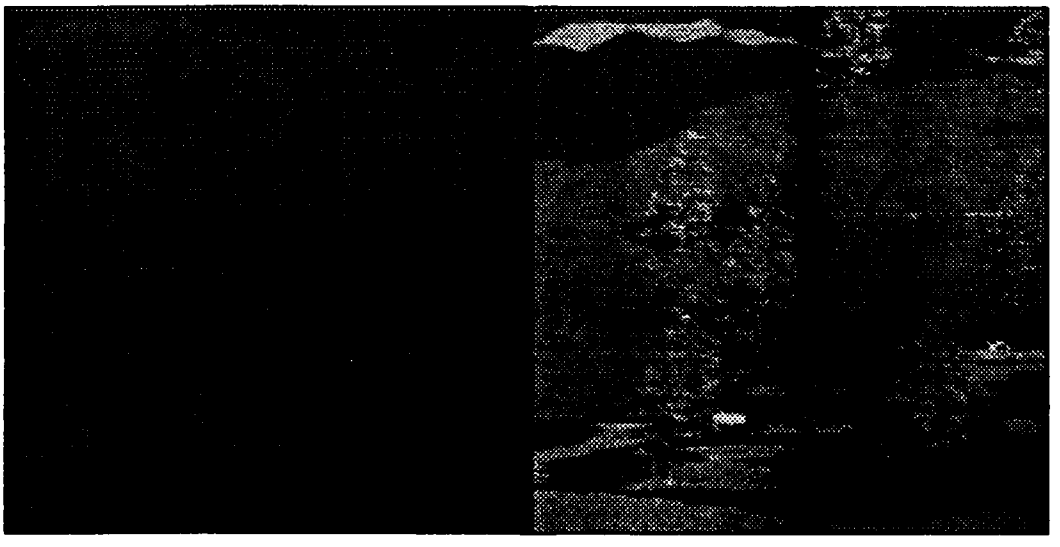

Figure 5. The synthetic IR images represent Fort Hunter-Liggett, California on 20 September 1989 at 0600 and 1500.

5. DATABASE CREATION AND SCENE GENERATION

5.1. Shaded relief

As a quick look quality check, the shaded relief program creates a top down gray-shaded relief image of the CIG's elevation grid data. The user inputs to this are grid file name, output image name, sun zenith, and sun azimuth. The output image of the shaded relief program is the CIG sector image format which was described in section 3.4. The sun zenith and azimuth determine how the elevation is shaded.

Since the image display routine on the Gould was hardware dependant, this routine could not be ported to view the output image. In order to view the image, it was converted to a format that could be displayed. The Application Visualization System (AVS), a software package available for the Titan, was used to view the resulting image. A program was written to convert the sector image to an AVS image for viewing. Since then a Motif Graphical User Interface (GUI) for the shaded relief program was developed. This interface allows the user to adjust all of the inputs described above, generate the image, and display the image on a 24-bit display. The user may also select points on the screen to obtain latitude and longitude information from the image. The GUI is written in C and simply outputs the mentioned parameters to a file that the shaded relief program reads in. The GUI is a C program shell that is written around the shaded relief FORTRAN program.

5.2. Terrain creation

The terrain creation program is the module that creates the CIG terrain database from the General Grid File (GGF). This is the same GGF that was used to generate the shaded relief image. This program creates three files: a region file, a node file, and an item file which are collectively known as the Common Database (COMDB). The region file is mainly a header file and contains such information as a region description, paths to item, node and texture file, number of texture maps used by region, which texture maps are used, center of the region in total arcseconds, size of the region in latitude and longitude, number of levels of detail (LOD), number of tree levels, and total number of nodes. A node is defined as a block of 5 x 5 grid data points. The node file defines the centroid, size, LOD ranges, pointers to sibling and children, FID's/SMC's, grid spacing, and pointers to the nodes geometry, color, and texture information for each node in the file. It also contains 64 one bit flags per node which describe such information as child pointer, sibling pointer, LOD and texture existence, shading type (curved, face, self -luminous), coloring type (vertex or polygon). These nodes are the building blocks of the database which is designed around the quadtree structure, that is, a tree structure that contains four children per parent which is illustrated in Figure 6.

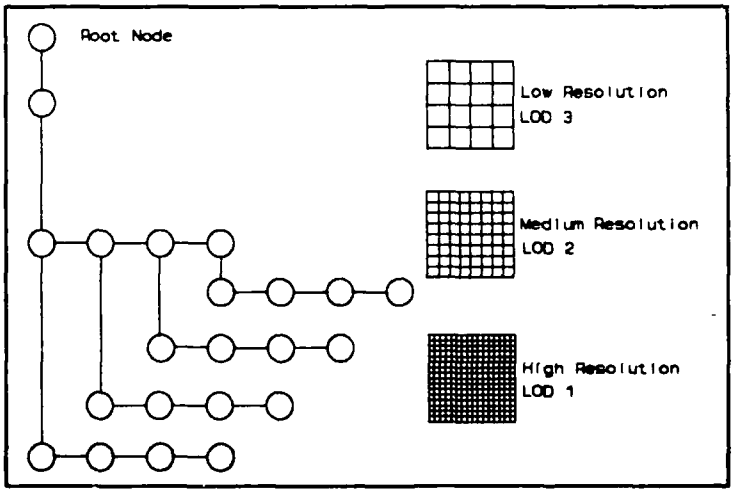

Figure 6. Quadtree structure for a database with three LOD's.

The item file contains the geometry information, color information, and texture information of the terrain. The geometry data is stored as relative offsets from one another. The only point in the database that is the true latitude and longitude position is the region centroid. The southwest corner is an offset from the region centroid and vertices on the west edge are relative to vertices immediately south. The other data columns are relative offsets from the vertices immediately west. The data are broken up into clusters or nodes of 5 x 5 grid posts. The database contains 3 levels of detail with the highest level of detail representing the highest resolution of the database. That is, all the data points. The next level of detail would represent every other point and the last would be every third point. A tree representing 3 LOD's is depicted on the previous page.

The color information in the database contains FID's which are indices into the CIG color LUT. A texture map number is associated with each node as well as (e,f) pairs which are row and column indexes into the texture map file. These pairs specify which texture element (texel) is assigned to the elevation vertices in the node.

Since the smallest element in the database is a node, the size of the GGF in longitude must be a multiple of the node size in longitude and the size of the region in latitude must be a multiple of the node size in latitude. Otherwise, this will result in an invalid quadtree. If the GGF is not the correct size, then when the COMDB is created, the file will be clipped to a multiple of the node size. As a result, some data will be lost on the east and north edges since the origin is the southwest corner. A file containing information such as northwest, southeast, and center of the database as well as grid spacing is created by the terrain creation program. This file is used to fit the color feature data to the terrain due to the data clipping.

5.3. Terrain coloring

The coloring routine uses the CIG color file, discussed in section 3.2, to color the terrain database. The parameter file output from the terrain creation program mentioned in the previous section is used to cut the raster file to the COMDB size. The fit program simply clips the north and east edges of the raster file based on the bounding coordinates obtained from the parameter file. The color file contains the FID/SMC data that are added to the COMDB. The FID's are used as indices into the CIG color LUT. The feature raster file is exactly the same size as the terrain database which produces a one for one mapping of a feature pixel to an elevation post.

5.4 Terrain texturing

The texture image data generated by the texture generator, discussed in section 3.4, was converted into a CIG texture file through a reformatter program that stored the texture images into a single file. The texture file contains 256x256 size maps plus 7 downsampled maps for varying LOD's (128x128, 64x64, 32x32, 16x16, 8x8, 4x4, 2x2). A texture translation file contains information which maps texture to terrain polygon. This file is built by reading the output file from the radiance model which contains FID's and their corresponding SMC's. The program builds the translation file and associates a texture map to each FID based on its surface material type (bare ground, medium vegetation, and forest canopy). Each record in the translation file contains an FID start and end, a map number, and an X and Y scale value. The X and Y scale value describes the amount of ground space that the texture map covers. Each pixel represents a specified amount of ground space in meters. Meters are converted to arcseconds and multiplied by 256 to obtain the amount of ground space the texture map covers. The terrain texturing program reads in this translation file and traverses the terrain database assigning texture maps based on their stored FID's. During image generation, the original color of the elevation vertex is modified by an additive method. Each texel value applied to the triangles are subtracted by 128 to remove the mean and then added to the value of the vertex color. With the subtraction of the mean, a texel value may become negative. If the value of the vertex is less then the absolute value of the texel color the resulting pixel color may be negative. Also, it is possible for a texel value to be added to a triangle color resulting in a value greater than 255. Either way the value will be clipped to a color in the range of 0 to 255.

5.5 Models

Both trees and targets are models in the CIG system. Models are supplied in the vertex/polygon format as described in section 3.5. These files are processed by a CIG program that converts them into the same format as the terrain database. That is, each model has a region, item, and node file associated with it. The models built are added to the scene through the image input file which specifies the model's positional information: latitude, longitude, altitude, yaw, pitch, and roll. The positions for the trees are obtained from a tree position file, which contain the positional information: latitude, longitude, and altitude at mean sea level. The targets positions are obtained through a point and click routine that allows the user to display a previously generated image without models and point to the position to add the model. This routine currently outputs the positional information to a file. The user then must input this information into the image input file through a standard editor. In the future, this display routine will modify the image input file to include the targets position. If the user wants to add additional trees, he may add them in this manner.

5.6 Image input file

The image input file drives the resulting output of the image generator. This file contains information that specifies the terrain region that will be rendered, image size, sun azimuth and zenith, horizontal and vertical FOV, ground, sky and haze color, and the sensor and model location information: latitude, longitude, altitude, yaw, pitch, and roll. Currently, a sky model does not exist that works with the scene renderer, so the sky and haze color must be specified by the user in RGB values. If there is any horizon in the rendered image it will contain the sky and haze color. The user also has the ability to toggle various flags on or off. The two most useful flags used are whether to output the depth file and whether to do texture processing. Even if the region has been textured, it does not have to be processed by the scene renderer.

5.7 Image Generation

All the routines discussed up to this point in section 5 build a database for perspective scene rendering. The rendering process consists of several steps that lead up to a final image. The steps involved consist of sending data through several processing routines which are the node processor, item processor, pretiler, tiler, and Z-buffer processor.

5.7.1 Node processor

The node processor has one purpose and that is to traverse the database and perform an FOV test on each node in the database. If the radius of the node is within the FOV then the node is processed further by being passed to the item processor, otherwise it is discarded.

5.7.2 Item Processor

The item processor performs several functions: triangle face normal calculations, polygon coloring/shading, vertices to viewpoint space conversions, polygons clipping to hither and yon plane. The hither and yon planes define a bounding volume and clip in 3-D space any triangles that are before (hither) or beyond (yon) this volume.

5.7.3 Pretiler

The pretiler receives data describing triangles in viewpoint space. The data for the triangles consist of vertex coordinates and color. If texture has been applied each vertex also has E,F texture indices associated with them. These are indices into the texture map associated with the triangle. The pretiler transforms this data into screen space coordinates through matrix multiplications, calculates color and depth gradient information of the triangle and clips the triangle to the screen.

In order for screen clipping to take place, the pretiler calculates a bounding box for each triangle which is simply the smallest rectangle into which the triangle will fit. At least one or as many as three of the triangle vertices will lie in the corner of the box. The bounding box is calculated by sorting the I's and J's of the triangle. The upper left corner of the box is $(I_{min}J_{min})$ and the lower right is $(I_{max}J_{max})$. There are four conditions that may occur in the clipping process. The first is the simplest, and that is if the area of the box does not intersect the area of the screen, then the triangle is not within the screen and no further processing is necessary. The second is both the triangle and the bounding box are completely within the screen boundaries which mean the triangle will be processed normally by the pretiler. The third and forth case is that the bounding box is partially on the screen, but the triangle may or may not also intersect the screen. These last two cases require some 'non-trivial' screen clipping. These conditions are depicted in Figure 7.

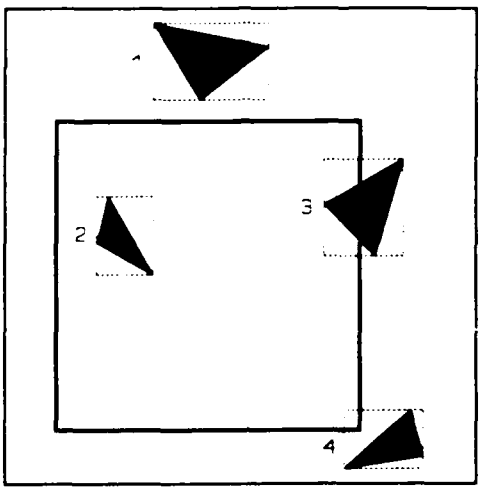

Figure 7. There are four cases in which triangles may be clipped to the screen.

5.7.4 Tiler

The tiler receives input from the pretiler which consists of triangle vertices described in screen coordinates along with the depth and color of each vertex. The tiler determines which pixels lie within the interior of the triangle and calculates the depth and color of the interior pixels by interpolating using the vertex information. If texture is present, it is calculated and applied to the pixels. The output of the tiler consists of the coordinates, depth, and color for each of the interior pixels of the triangle processed. This information is passed to the final processing step, the z-buffer.

5.7.5 Z-Buffer algorithm

The z-buffer algorithm is the final process before the image rendering is complete. This algorithm is used to determine whether one object obscures another. The z-buffer stores z values or depths for each pixel. The first step is to initialize the depth buffer to the maximum z value and the image buffer to the background pixel value. The background pixel value is determined in the image input file by the values set for ground, haze and sky color. As the tiler determines the pixels within the triangle and the depth of each pixel, if the depth of the pixel currently being processed is less than the value already stored in the z-buffer, then that polygon is closer to the viewpoint at that pixel than the previous one recorded in the z-buffer. The depth and intensity of this pixel replaces the current value stored in the z-buffer. If the depth of the pixel is farther away than the one stored in the z-buffer, then it is discarded. When all the triangles in the FOV have been processed, the z-buffer will contain the smallest z values for each pixel and the image buffer will contain the intensities corresponding to those z values. The buffers are then stored into two files: a sector image file and a depth file.

5.8 Image display

The image rendered is stored in the sector image format and has a corresponding depth file which contains depth values for each pixel in the image file. The depth stored is actually stored as the inverse depth or 1/depth. The image is displayed by means of a program written using X-windows. Besides strictly displaying the image, the program also allows the user to get positional information from the screen through the use of a mouse. This information is processed by converting screen pixel coordinates to world coordinates using the depth information and sensor location. A new image displayer written using X-motif is currently being developed and will allow the user to display the IR image in pseudo color using a default color LUT or a user-defined color LUT. The user will also be able to display the image as a range map with the closest pixels colored black and increasing in intensity to white as pixels are further from the sensor viewpoint.

6. FUTURE WORK

6.1 Image sequencer

Future work will consist of an image sequence generator which could be stored to an optical disk or VCR for later previewing. Since the image generator works with color LUT and the terrain geometry does not change depending on time of day, a user could run a weeks worth of data through the thermal and radiance models (that is, 24 hours a day for 7 days) and come up with 168 color LUT's for the terrain. The image generator could be modified to generate an image for each LUT. These images could be displayed in sequence to see how the terrain radiances change over a weeks time period. There is, however, a problem with models. The trees and targets do not work on LUT since the actual color is already stored in the item file. This would mean that models would have to be generated for each of the 168 runs which could take up a lot of disk space. If done this way, the image input file would have to reference 168 representations of the same model. This is a definite waste of space because all the models have the same geometry, but different colors. A solution to this problem would be to separate the geometry data and the color data into two separate files.

6.2 Graphical User Interface (GUI)

A GUI for the shaded relief is already available for the Titan. An image viewer is currently being developed for the SG and Titan. A GUI to modify parameters in the image input file and to run the entire database creation and scene rendering process is currently under development.

7. ACKNOWLEDGEMENTS

The author wishes to thank the following participants in the SWOE project: US Army Cold Regions Research and Engineering Laboratory, US Army Engineer Waterways Experiment Station, US Air Force Phillips Laboratory/Geophysics Directorate, SPARTA, and Aerodyne Research Incorporated.